Analyzing Acoustic Imagery in 3-D: A Case Study

A.R. Haas
Logicon, Inc. Herndon, VA

S. Ziegler
Mississippi State University, Starkville, MS

P.P. Gruzinskas
Naval Oceanographic Office, Stennis Space Center, MS

***Abstract*—The U.S. Navy has always had an acute interest in the seafloor. It frames their battlespace, and knowledge of its features and characteristics can lead to tactical advantage. Modern survey technology, combined with precise positioning systems, has provided oceanographers more data than can be comprehended effectively. The task of scientific visualization is to represent the data both accurately and effectively. The application that is applied to the data must communicate the essential features within the data to the user. In the case of the seafloor, the ability to render the terrain in three dimensions (3-D) has not only made the data easier to interpret but provides a framework for intuitive displays of other data within the context of the submarine bottom. This case study will demonstrate effective real-time rendering of gridded bathymetry, using a level of detail implementation to reduce the polygon count and provide a tool that is both efficient and portable. A navigable plan-view, graticule, and depth probe help to quantify the data within the application, which is primarily mouse-driven.**

The second part of this case study will deal with applying acoustic imagery to the surface of the seafloor as a texture. High-frequency towed sensors are producing ultra-high-resolution imagery. These data can be manipulated rather painlessly in two dimensions (2-D); however, rendering a 3-D-textured surface in real time can tax even the strongest hardware. This application was designed to support mine warfare and mine countermeasures. To achieve interactivity with the data, a technique called 3-D clip-mapping or clip-texturing is applied. It leverages specialized hardware, which uses dedicated texture memory and can manage the varying resolutions created to facilitate interactive frame rates. Although some effort was expended to produce this application in OpenGL, it was decided to leverage the extensive development efforts devoted to Silicon Graphics, Inc.'s Performer product to accomplish this task. The Naval Oceanographic Office Major Shared Resource Center Visualization Center has made significant accomplishments toward the display of other ocean parameters within our 3-D ocean environment, such as ocean circulation, temperature, and bioluminescence. This case study will address only the applications developed for the bottom geometry, which, because of their efficiency, enable the incorporation of additional environmental information.

I. INTRODUCTION

The Navy's growing interest in acoustic backscatter imagery has been complemented by its proficiency in collecting, processing, and analyzing these extremely valuable data. For many years the Navy has collected these data in support of hydrographic survey operations. Side-scan sonar is unsurpassed at detecting underwater hazards to navigation.

In the early 90's geophysicists at the Naval Oceanographic Office (NAVOCEANO) began processing the raw sonar returns from the low-frequency hull-mounted multibeam bathymetric systems used for deep-water surveys, to extract backscatter information [1]. To create this acoustic imagery, amplitude information was extracted from the raw sonar return and converted into an acoustic image. The resolution of the mosaicked imagery extracted from these low-frequency hull-mounted systems was on the order of 50 m and provided valuable information regarding the acoustic properties of the seafloor. The stacked scan lines offered a higher resolution view of the data, but analysis of this data did not provide a truly geo-referenced view. These low-frequency deep-water systems had limitations working in shallow water. Since then, the Navy mission has moved into shallow littoral zones.

To meet this new mission, scientists at NAVOCEANO have been involved in collecting, processing, and analyzing acoustic imagery from various sources, to support a wide spectrum of tactical operations. These operations include, but are not limited to, hydrographic hazard detection, mine warfare, mine countermeasures, acoustic modeling, salvage operations, and cable laydown.

Some of the systems the office employs to reveal the texture of the seafloor include low-frequency hull-mounted, low- and high-frequency towed, and CHIRP (swept frequency). The original analog systems have been replaced with digital systems, which facilitated near-real-time display of these crucial data.

These data are processed and displayed by both commercial off-the-shelf (COTS) and custom software applications that use algorithms for filtering, object detection/clutter analysis, and bottom characterization. Custom products are generated once the data has been processed and controlled for quality. The resolution of the mosaics produced from the high-frequency towed system discussed here is 75 cm, but the software application outlined in this case study will deal with even higher resolution imagery.

The underlying bathymetry used is .1 arc minute, or approximately 180-m resolution. The Klein 5000 survey system leverages a 455-kHz signal and an operator-selectable pulse length (50 to 200 μsec) to collect high-resolution backscatter data. Cross-track resolutions from 7.5 to 15 cm

can be achieved using these systems. The speed of the collection vessel ultimately determines the resolution of the final mosaicked, or gridded dataset. The Major Shared Resource Center Visualization Center is working with NAVOCEANO's Image Processing Lab to display these data as a texture that when draped over the 3-D terrain can offer additional insight into the ocean floor and its related processes.

The display of 2-D pixels, even high-resolution imagery, can be accomplished rather easily on standard desktop workstations. The interactive display of these data as a texture that is then applied to a 3-D surface (bathymetry), however, requires specialized hardware and software techniques. A texture pixel is mapped to a 3-D space. The hardware required is part of Silicon Graphics Inc.'s (SGI's) Onyx2 graphics architecture, which is called the Infinite Reality2 graphics pipeline. The raster managers, or texture memory, are dedicated to the storage and rapid retrieval/display (paging) of textures. The software used is the SGI Performer product, which contains a 3-D clip-mapping technique [2]. This technique uses a dynamic level of detail scheme to efficiently render the full-resolution imagery in the user's near field, while using lower resolution (over-sampled) versions of the gridded data in the user's intermediate and far fields of view. This allows for interactive (greater 30 frames/sec) roaming of these large datasets in 3-D space. As commodity desktop architectures improve, in part due to the evolution of the 3-D computer gaming industry, this type of memory will be common on standard 3-D graphics cards.

II. BATHYMETRY SOFTWARE

The size of both modeled and measured data in current ocean studies challenges a workstation's ability to visualize data at interactive speed. Usual methods of visualization relied on heavy batch-style post-processing of the data into image files that can be played back as movies or plotted and studied for future use. Interactive visualization has been traditionally done with COTS packages that statically construct geometry of the entire data surface and then render this geometry as one piece. Graphics systems of mid-end workstations often cannot render the geometry fast enough to be interactive.

Data from ocean studies is comprised of bathymetry and bathymetric texture that is often too large to be visualized directly. In this paper we describe two software packages developed at NAVOCEANO that perform interactive rendering of large-scale 3-D bathymetric terrains and render large-scale 2-D texture datasets mapped to 3-D terrains. The software uses demand-paging of graphics, memory, and I/O resources to accommodate very large data spaces. This enables the exploration of ocean data sets anywhere across the earth using the memory and graphics present on today's midrange desktop workstations.

A. CharterExplorer

CharterExplorer is a software package developed at NAVOCEANO that lets a user roam across large areas of bathymetry. The program reads bathymetry data from the Charter file format (used by the Seafloor Data Bases Branch at NAVOCEANO). The data are reconstructed into a 3-D surface of the seafloor. The reconstruction takes place "on-the-fly," dynamically optimizing the level-of-detail of the areas within the user's current sight. This dynamic approach is required to navigate data spaces that would bottleneck the graphics systems.

The Level-of-Detail (LOD) rendering used in CharterExplorer actively manages the scene's geometry, so interactive rates are maintained. Only geometry for the terrain within the user's field of view is constructed. The construction uses multiple LOD of the bathymetric data to ensure that high-resolution geometry is conserved for terrain features near the user.

LOD rendering is accomplished using a feature called Mip-mapping. Mip-mapping is a conventional feature of texture mapping hardware that uses a low-resolution texture when shading far away objects, and high resolution when shading objects up close. CharterExplorer applies this principle to bathymetry data by reducing the data into a fixed number of mip-map levels, N. The levels range from 0 – N-1, with level 0 containing the original full-size data and level N-1 containing data reduced by a factor $2^{-(N-1)}$, effectively reducing the data by 50% in each level. N is chosen to give a minimum resolution typically in the range of (50 x 50) to (100 x 100).

CharterExplorer divides the area into fixed-size tile units. Each tile can be quickly checked if all or part of it lies within the user's field of view. The mip-map level of each tile is a function of the tile's distance from the viewer. The user can stretch or contract this function as the program is running via a couple of keystrokes. This has the effect of using lower resolution mip-map data across the horizon (stretching), and using higher resolution data across the same horizon (contracting). CharterExplorer continuously computes the tiles and their levels within the mip-map as the user's field-of-view changes.

Fig. 1 shows CharterExplorer applied to a 3600x2400 global bathymetry dataset (taken from a 1/10 degree Parallel Ocean Program (POP) model). The bathymetry closest to the viewer is rendered at a higher resolution mip-map level than the farther bathymetry. This enables large spaces of data to be explored at interactive speeds.

CharterExplorer allows the user to probe the rendered data by clicking the mouse over the desired location. This action causes a marker to be placed on the seafloor of the clicked location. The marker is annotated with longitude, latitude, and depth. The probe may be dragged across the bathymetry for continuous feedback.

III. ACOUSTIC TEXTURE MAPPING

Sonar texture data are mapped onto the 3-D surface bathymetry using a hardware technique called Clip-mapping. Clip-mapping is a feature of a graphics hardware that performs texture mapping across very large 2-D textures. In this case, the texture comprises pre-processed scan-lines of sonar imagery data.

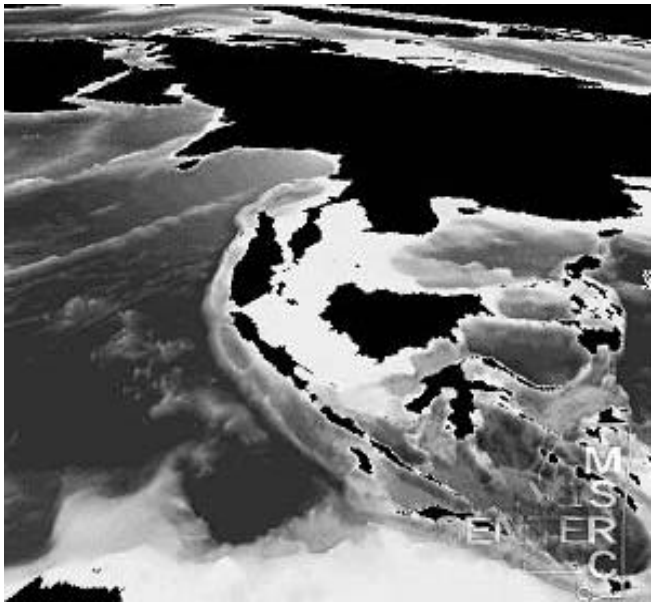

Fig. 1. View from CharterExplorer looking northeast from Australia across a 3600x2400 global ocean bathy dataset.

In implementing a real-time navigation application that would handle large textures using clip-mapping, we decided that IRIS Performer was a suitable option. It has the advantage of being abstracted to a level above OpenGL, so writing code should be easier. Another advantage is that Performer is finely tuned to operate on SGI graphics hardware, so applications should have greater performance, especially on higher-end systems. Usually, this hardware-specific approach is a disadvantage because of the loss of portability; however, the clip-mapping hardware is currently available only on SGI systems. Thus, an implementation in OpenGL is just as platform-limited as a Performer counterpart.

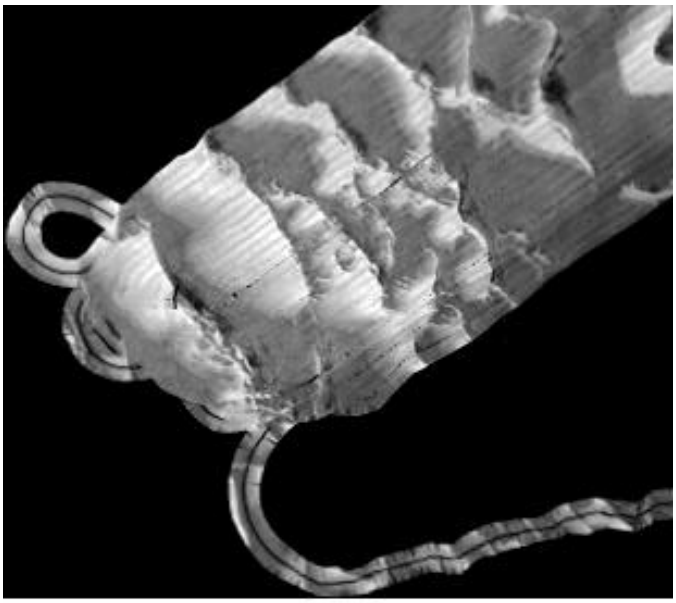

Fig. 2. Top down view of complete 8192x8192 acoustic texture space. The lowest level of texture resolution in the clip-map is used.

A. Performer Overview

Performer consists of several libraries from which an application can be built. At the lowest level, the "libpr" library provides basic rendering functions. One layer above this is the "libpf" library, which provides multiprocessed rendering of a database of objects defined by a scene. Other libraries include "libpfdu" for managing object and file formats and "libpfui" for user interface functions. We used the "libpf" library, since it provided us with a high-performance interface and a level of abstraction suitable for rapid development of this specific application.

The heart of a "libpf" Performer application is the scene graph. A scene graph is a directed acyclic graph of the elements of some virtual scene. Objects may be added or removed from the scene graph during program execution. The scene graph is given to the Performer rendering engine, where it is drawn in a graphics window. Performer provides all of the functions necessary to create and modify the scene graph, as well as pass the scene graph to the rendering engine.

Multiprocessing in a Performer application provides a performance advantage when the application is on a system with more than one processor. Performer uses multi-processing by dividing an executable into three phases: application, culling, and drawing. The application phase is where the program handles computations, user input, and other miscellaneous tasks. The culling phase removes any objects within the scene graph that are outside of the field of view. The drawing phase is where the remaining objects are drawn in the graphics window. On a system with multiple processors, these phases are pipelined such that while the first frame is being drawn on one processor, the second frame is being culled, and the third frame is in the application phase.

B. Implementation

In implementing our clip-texturing application, we divided it into five steps:

1. Create a Basic Performer "Plane Navigation" Application
2. Add Multiprocessing Capability
3. Add a Simple Texture
4. Change the Texture to a Clip-Texture
5. Add Topography Information

With these steps, we could start with a basic application and add features until we achieved our goal.

STEP 1: Create a Basic Performer "Plane Navigation" Application

The first step involved adding the components necessary for any "libpf" Performer Application. The scene graph consists of a GeoSet, which defines the geometry, and a GeoState, which defines the properties of that geometry (color, texture, etc.). The GeoSet is simply a plane, defined as a rectangle covering the same area as the texture. The GeoState defines the rectangle's area initially to be white.

The mouse was used to navigate around the plane. Pressing a button would fly forward or backward, and

moving the mouse would steer. Since the "libpfui" library is not used in the interface, we used the X-toolkit Intrinsics to access the mouse and keyboard. An asynchronous callback function checks the status of the mouse and keyboard and updates global state variables based on the input. A separate function updates the viewer position based on these global state variables. Finally, the scene is rendered based on the new viewpoint. This loop occurs every frame.

STEP 2: Add Multiprocessing Capability

Adding multiprocessing requires little work in a "libpf" Performer application. A single function is used to inform Performer how to set up the multiprocessing. One may configure performer to use a specific number of processors or allow Performer to choose a suitable configuration based on the hardware. We allowed Performer to choose the configuration.

Our application also used global state variables. This is a problem in multiprocessing applications, since each process has its own copy of each global variable, independent of the others. To solve this problem, Performer established a shared memory arena and provides functions to allocate memory in the shared arena. These state variables were put in the shared arena, where all processes had simultaneous access.

STEP 3: Add a Simple Texture

To add a texture to the plane, one enables texturing in the GeoState associated with the plane. A texture node is attached to the GeoState, which is a simple step in Performer.

STEP 4: Change the Texture to a Clip-Texture

Performer treats clip-textures just like textures, so associating it with the GeoState is all that is required to attach it to the geometry. However, additional information is required, since the clip-texture can be updated to change its center, where it uses the highest resolution possible. This information can be included in a configuration file that can automatically be read by a Performer application.

Also, a multiprocessing clip-texture node (MPClipTexture) must be attached to the clip-texture. The MPClipTexture node is then added to the scene graph as a parent to the geometry node. Performer provides the functions necessary to update the MPClipTexture's center. The center is updated based on the user's viewpoint, such that the highest resolution in the texture is where the user is looking.

STEP 5: Add Topography Information

Finally, a standard format for loading topography information and building a grid must be defined. Instead of the plane, this grid will be the 3-D surface bathymetry onto which the texture will be mapped. The CHRTR format used by CharterExplorer was selected since the application was directed toward texturing sonar imagery onto the seafloor. Also, this requires modifying the collision detection to prevent the user from flying through the terrain.

Fig. 3 shows the application of the Performer Clip-Texturing technique to the 8192x8192 acoustic imagery shown in Fig. 1. The texture has been overlaid with an exaggerated coarse bathymetry. The clip-texturing allows the user to interactively roam across texture sizes that exceed the hardware's direct texture mapping capability. The small bump at the lower center portion of the figure is a minelike device seen at the highest resolution.

IV. FUTURE WORK

As resolutions of both bathymetry and acoustic imagery inevitably increase, fusion of the level of detail algorithm used for the bathymetric surface, and the clip-mapping technique applied to the imagery will have to be accomplished. Another promising technique for the analysis of acoustic imagery in 3-D involves the creation of a bump map from the acoustic image, which can then be rendered in 3-D using Phong shading. Most importantly, the porting of these techniques to commodity desktops and even laptop architectures will be pursued as the requirement to efficiently provide digital data to the fleet becomes increasingly important.

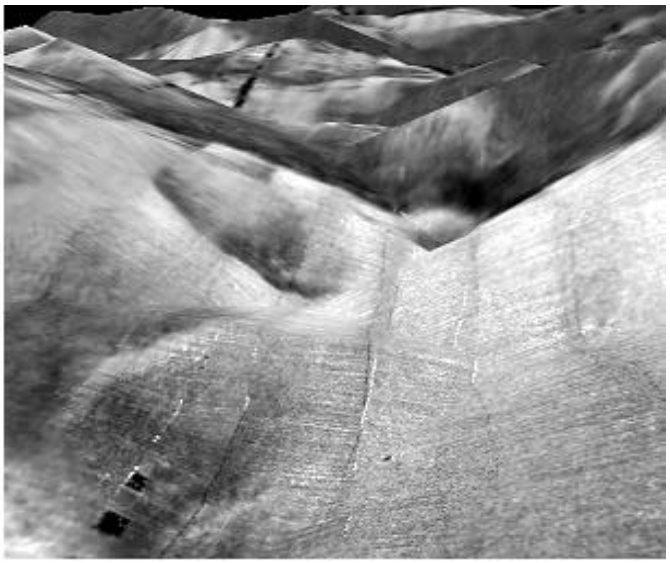

Fig. 3. Zoomed view of 8192x8192 acoustic texture in Fig. 2. The texture is laid across an exaggerated topography. The clip-mapping technique uses a clipped version of high-resolution texture near the viewer. Textures that lie outside the high-resolution clip-map are progressively mapped to lower resolutions.

REFERENCES

[1] S.C. Lingsch and C.S. Robinson, "Processing, Presentation, and Data Basing of Acoustic Imagery," *Oceans '95 MTS/IEEE Conference Proceedings*, pp. 1582-1591, 1995.

[2] J. Rohlf and J. Helman, "IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics," *Computer Graphics (SIGGRAPH '94 Proceedings)*, pp. 381-394, 1994.